自己管理データベース：自動健全性監視およびアラート

概要

企業のデータベースのサイズと数は増加の一途をたどっており、結果としてシステム管理がますます複雑になってきています。Oracle Database10*g* では、管理の簡素化、効率化およびシステム管理コストの低減を実現する多数の自己管理機能を導入しました。アラート管理は、Oracle Database10*g* で一新された領域の 1 つであり、様々なサブシステムでの事前問題解決、障害管理およびタイムリーなアラートの自動生成によりデータベース管理者を支援します。

このホワイト・ペーパーでは、**サーバー生成アラート**と呼ばれる Oracle Database の健全性を監視する新しい方法について、関連のアーキテクチャ・コンポーネントと共に説明します。また、Enterprise Manager（EM）の高度なアラート伝播フレームワーク、およびそのフレームワークとサーバー生成アラートとの統合についても説明します。EM と効率的な**サーバー生成アラート**システムとを組み合わせることにより、QoS（quality of service）に焦点を当てた完全なシステム監視ソリューションが完成しました。

注意を必要とする理由

人生のあらゆる問題に対する解答は、42 [Adams,1]であると聞いたことがありませんか。つまり、アラートのしきい値をすべて魔法の数 42 に設定し、後は放っておいてもよいのです。さらに、Oracle を使用すれば、この処理を自己管理形式で自動的に実行できます。

冗談はさておき、データベース管理者（DBA）は、データベース・システムの健全性の監視、ボトルネックの特定およびシステム・パフォーマンスの改善に、多くの時間を費やしています。パフォーマンスなどデータベース中心の問題の検出は、適度なサービス・レベルでシステムを継続的に運用する上で不可欠です。DBA は、急を要する問題についてタイムリーに通知を受け、すばやく診断し、問題解決対策を取る必要があります。責任感の強い DBA であれば、事後対応で済ませることは決してしません。システムの不具合が年に*何日もあってはたまりません。***サーバー生成アラート**は、必要な監視機能をほとんどオーバーヘッドをかけずに提供します。

市販の監視システムが抱える問題

現在市販されているデータベース健全性監視ソリューションは、多くの問題を抱えています。ここでは、Oracle Database10g の自動健全性監視機能が開発された背景ともなった、そのような問題のいくつかについて説明します。

観察者が観察対象に影響を与える（巨視的）

Werner Heisenberg（1901-1976）は、20 世紀初頭に提唱された、量子物理学の近代理論および原理の先駆者となったドイツの物理学者です。**ハイゼンベルグの不確定原理**が最初に発表されたのは、1927 年 2 月でした。*この原理では、「粒子または物体の位置または速度を測定しようとする試みは、物質、少なくとも亜原子のスケールでは目立つものである」と述べられています。*つまり、物質の正確な位置と正確な速度を同時に測定することは不可能である、という意味です。

それでは、データベース健全性監視と量子物理学とは何か関係があるのでしょうか。実際には多くの共通点があります。この先を続けてお読みください。

不確定原理 – 必要な迂回

光は、光量子と呼ばれるエネルギーの粒で構成されています。粒子の位置と速度を測定するには、まず、物質または粒子に光をあて、その後、粒子または物質から反射してきた光を測定する必要があります。巨視的なスケールでは、光量子が物質に与える影響は重要ではありません。

しかし、亜原子のスケールでは、光量子が亜原子の粒に当たると亜原子が大きく移動するため、位置を正確に測定できたとしても、粒子の速度は変化します。亜原子の粒子の位置を測定することによって、速度に関してこれまでに収集した情報がすべて無駄になってしまいます。言い換えれば、**観察者が観察対象に影響を与える**のです。

データベース健全性監視と不確定原理

不確定原理は、日常生活のほとんどすべての分野に当てはまります。サードパーティ製のデータベース健全性監視の場合には、不確定原理の影響は、光量子の場合の亜原子スケールとはまったく異なり、**巨視的なスケール**で表れます。サードパーティ製の監視ソリューションが監視対象のシステムに重大なレベルで影響を与えることから、データベース健全性監視の場合には、不確定原理は巨視的レベルにさえも当てはまることがわかります。

このことから導かれる結論は、**観察プロセス**が**観察対象**（データベース）に与えるどのような影響も観察対象の健全性に影響を与えるレベルであってはならない、ということです。ここで「観察対象に影響を与える」ことに関連するのは、リソースのオーバーヘッドだけではありません。発生するリソースの競合も関連します。リソースの競合が激化すると、観察対象のシステムの機能が低下し、事実上使用不能になります。

たとえば、具合の悪い人が病気の原因を調べるために来院したとします。患者の容態を観て、医者は生命に関わる重要な統計がすべて標準であることを確認するために*採血*しようと決めます。*その医者にしかわからない理由のために、15 分おきに採血されます。*このように過剰にリソースを使用すると、結果的に患者を死に至らしめます。新しい血球を作る人体の機能が採血の量に追いつかないためです。**観察者（医者）が観察対象（患者）に影響を与えています。**ここでは、オーバーヘッドと過剰なリソース使用の違いは区別されています。

過剰または非効率的なデータベース健全性監視は、対象となるデータベースおよびホストにおいて、容認できないレベルのリソース使用と競合を招きます。監視中、これらの対象に多大な**影響を与えます**。対象が受ける影響は、アプリケーションまたは基盤システム・コンポーネントの機能停止につながるレベルです。*サーバー生成アラートは、この問題を解決するソリューションを提供します。サーバー生成アラートによる観察の影響は亜原子スケールのみで測定可能です。*

過剰なオーバーヘッド

これまで多くの監視ソリューションでは、SQL を使用してデータベース測定および統計の値を ping し、データベースの健全性を監視していました。統計のために永続的にデータベースを ping するので（**データ・プル**）、この方法では高い負荷がかかります。多大なシステム・オーバーヘッドを課すパフォーマンス監視ソリューションに出会うことも珍しくありません。オラクル社では、1%を超えるオーバーヘッドはすべて**大きい**と考えています。

最近は、Oracle System Global Area（SGA）に直接アタッチする方法を採用する新種の監視ソリューションも販売されています。このソリューションのセールス・ポイントは、Oracle SGA に直接アタッチする際、SQL レイヤーによって課されるオーバーヘッドを排除できることです。確かに、多くの場合は排除できます。

しかし、直接 SGA にアタッチするソリューションは***オーバーヘッドが皆無***の状態で動作するわけではありません。たとえ SQL レイヤーを使用せず、多くの SQL ベースのオーバーヘッドをシステムに課すとしても、直接 SGA にアタッチする監視ソリューションは監視対象のシステムの CPU を大量に消費します。このような監視では、Oracle の共有メモリー領域内の各種 C 構造を往来し、関連の情報を頻繁に**サンプリング**する必要があるため、テリトリーの問題が伴います。Oracle のメモリー構造内の値が補助的に再びサンプリングされることも珍しくありません。

設定の複雑さとカスタマイズの必要性

前にビデオの設定をしたときのことを思い出してください。点滅する 12:00 を消し、正確な時刻を設定し、やもめ男の次のエピソードを録画すべく奮闘しながら、気が狂いそうになったのではないでしょうか。取扱説明書があっても、です。今こそ、Tivo ボックスに投資するときです。おっと、話が横にそれました。

今日市販されている多くのシステム監視ソリューションでは、設定に多くの時間と労力が必要です。これまで、監視システムを販売するサードパーティ・ベンダーが、少なくとも 1 週間分のコンサルティング時間を監視環境の設定に費やす例を多く見てきました。

データベース健全性監視の設定に伴う複雑さには、一般に、次の特徴があります。

- サポートするスクリプトが多すぎる
- スキーマ・オブジェクト（表、索引、ビュー、シノニムなど）が多すぎる
- デプロイメントに柔軟性がない（手作業による構成やタスクの繰返しが多すぎる）
- 拡張性に欠ける（大量配置のパフォーマンスが悪い）

問題検出から診断および解決への推移がない

もしあなた、もしくはあなたの知人が割れるような頭痛に見舞われたら、医者に行き実際の原因を特定してもらうでしょう。その医者が 15 分ごとに採血しないことを願いますが。適量の鎮痛剤を服用すれば症状が緩和されるかもしれません。しかし症状が続く場合には、実際の原因を究明する必要があります。

同様に、パフォーマンス監視でも、問題があると通知するだけでは不十分です。状況や問題の検出だけでなく、見つかった問題の正確な診断および解決につながる適切な情報の提供が必要です。結論としては、監視ソリューションは症状の治療だけでなく、根底にあるパフォーマンスまたは設定の問題の診断および治療を促進する必要があります。

Oracle Database10g の自動健全性監視の利点:サーバー生成アラート

Oracle Database10g の**サーバー生成アラート**には、多数の利点があります。ここでは、効率的で有意義なデータベース健全性監視に貢献する主な要因を概説します。サーバー生成アラートは、データベース健全性情報をユーザーに提供する Oracle の次世代アラートテクノロジです。

観察者が観察対象に影響を与える（亜原子的）

サーバー生成アラートの最大の利点は何でしょうか。まず、より多くのシステム・リソースを、ビジネスおよび企業に収益をもたらす有益な作業の遂行に利用できることです。またサーバー生成アラートは、対象となるホストおよびデータベースで利用可能な貴重なリソースに、まったく競合を起こしません。サードパーティ製のソリューションと比べ、Oracle Database10g の監視によるサーバー側の影響は**亜原子的**です。その理由は、*Oracle Database10g 自体が、監視対象となる物質または粒子であるためです。*

きわめて低いオーバーヘッド

アラートを生成および配信するオーバーヘッドは、必要なすべての統計の収集コストを含め、構成されているシステム・リソースの 0.1%未満です。これは、多くのサードパーティ製監視ソリューションによって課されるオーバーヘッドとは対照的です。オーバーヘッドがこれほど低い理由は、監視機能が組込みであるためです。データベース健全性情報を収集するために、SQL 実行や、各種メモリー構造をスキャンして頻繁にサンプリングの必要がありません。

サーバー生成アラートは、ジャスト・イン・タイム方式です。さらに、Oracle Database10g の自動パフォーマンス監視システムが搭載する長年のシステム・パフォーマンス最適化経験により、サーバー生成アラートおよび 10*g* アドバイザは、真の問題、さらには関連の診断および解決を正確に検出する高い可能性を秘めています。

効率的なデータ・プッシュ

測定のデータ値が短い間隔で要求（ping）される従来のデータ・プル方式と異なり、**プッシュ方式**では、問題が検出された場合**のみ**測定のデータ値が提供されます。この処理が可能となるのは、監視機能が Oracle 実行可能プログラムに組み込まれており、SQL を使用した外部のパフォーマンス・モニターではないためです。問題状況が検出されると、必要なアラートが生成され、詳細キューにプッシュされます。このキューは、データベース検出プロセスで Enterprise Manager によって自動的に発行され、必要なアラートが問題のデータベースに提供されます。

たとえば、USER_WAIT_TIME_PCT （ユーザー・セッションの待機時間率）が指定のしきい値を超えたことが検出されると、関連の通知が送信されます。このように、データベースを定期的に ping して数百に及ぶ測定や統計の現在値を取得することで、パフォーマンス問題の有無を確認する必要はまったくありません。これにより、貴重なシステム・リソースを節約でき、ビジネスの収益に直接貢献する機能に役立てることができます。

最小の構成

前述のように、サーバー生成アラートは、Oracle Database10g のインストールと同時に利用可能になり、追加の構成または設定が必要ありません。サーバー生成アラートは Oracle カーネルの一部であり、必要な箇所にはデフォルトのしきい値があらかじめ設定されています。今後のリリースでは、システム・ワークロードに基づいて、多くのしきい値を意味のある適切な数値にデフォルトで設定する予定です。Oracle Enterprise Manager 10*g* Database Control はインストール時にすでにデフォルト設定されており、アラートが Home Page で伝播および通信されています。このように、DBA は、Oracle Database10g の自動化され、オーバーヘッドが低く、デフォルト値がすでに設定されている健全性監視およびアラートを即時に利用できます。考えられる最高のソリューションではないでしょうか。

問題検出から診断および解決へのシームレスな推移

自己管理型データベース Oracle Database10g は、問題検出からタイムリーな診断および解決への推移を容易にする変更およびアドバイザ・インフラストラクチャを備えています。変更メカニズムはアドバイザ・フレームワークと強く統合されており、コンテキスト・センシティブな問題診断、さらには関連する解決方法を、アドバイザを使用して生成できます。

Oracle Database10g の自動データベース健全性監視

Oracle Database10g は、組込みで、オーバーヘッドが低く、必要な設定の少ない健全性監視機能を提供することで、高度なデータベース・パフォーマンス監視を実現します。データベース・インストールの一部である自動健全性監視は、すべてのサーバー・コンポーネントに共通のインフラストラクチャを備えており、通知および提案を事前および事後に外部クライアントに配信します。

高度なインフラストラクチャ

Oracle Database10g は、管理性に焦点を当てた初の RDBMS です。この取組みの一環として、自己管理、自己診断および自己チューニングを行うデータベースの頂点を極める画期的な機能をサポートする高度なインフラストラクチャが構築されました。新しいバックグラウンド・プロセス MMON、および Automatic Workload Repository（AWR）、Automatic Database Diagnostic Monitor（ADDM）および Active Sessions History（ASH）などその他各種のコンポーネントが自動監視の基礎的要素となっています。

MMON

Oracle Database10g では、サーバー内のすべての自動管理を処理する専用の管理バックグラウンド・プロセス MMON を導入しました。データベース・サーバーの各コンポーネントは、MMON を使用して監視アクションの定期実行をスケジュールできます。問題を検出したコンポーネントは、修正アクションがサーバーによって自動的に実行されるようスケジュールするか、ユーザーがアクションを実行するようアラート・メッセージを生成します。

同様に、フォアグラウンド・プロセス（サーバー・プロセスとも呼ばれます）が異常な状況を発見した場合にも、MMON によって実行される緊急アクションを起動できます。次にそのアクションによってメッセージが生成され、ユーザーに送信されます。これらのアラート・メッセージはどちらの場合も、信頼性が高くタイムリーな方法でユーザーにプッシュされます。アラート・メッセージには、問題の説明と修正方法に関するアドバイス（適宜）が記載されています。

また MMON は測定（システムによって収集された統計の導出値）をサーバーの組込みリポジトリに定期的にフラッシュし、それらの値の履歴を保持します。このような測定は、DBA にとっては目新しいものではありません。*バッファ・キャッシュ・ヒット率*という言葉ならよくご存知のことと思います。Oracle Database10g によって収集される測定は、前のバージョンで Enterprise Manager によってポーリングされていた値のスーパーセットです。

AUTOMATIC WORKLOAD REPOSITORY（AWR）

自己管理コンポーネントが問題の検出、診断および解決のサイクルを行う際、これらのアクションの正確さは、包括的なシステム・パフォーマンス統計が利用できるかどうかに大きく依存します。パフォーマンス統計を効率的に取得および保持する新しいサーバー組込み機能 AWR は、管理および監視インフラストラクチャに統合されています。AWR はデータベースの低オーバーヘッド・データ・ウェアハウスであり、自動的に維持されます。*スナップショットは、特定の時点で収集された AWR に格納された測定、高負荷 SQL およびホット・オブジェクトの集大成です。*

ASH

アクティブなセッションの履歴は、待機イベントと SQL 情報を伴うアクティブなセッションのインメモリー表現です。この履歴は、ディスク、AWR へのフラッシュ前に 30 分間循環バッファに保持されます。ASH は Oracle に組み込まれ、直接 SGA にアタッチするパフォーマンス・データ・コレクタです。

アドバイザ

アドバイザは、サーバー・セントリック・モジュールで、推奨事項を提供し、レルム内で特定のデータベース・サブコンポーネントのリソース使用率およびパフォーマンスの向上を実現します。アドバイザは、発生した問題に関連する豊富なコンテキスト・センシティブ情報を DBA に提供します。アドバイザは、アラートメカニズムを補完します。アドバイザは、現在のインメモリー・データだけでなく履歴データを参照して、推奨事項を作成します。アドバイザは、他の方法では取得できない重要なチューニング情報を提供するため、きわめて強力です。

ADDM、SQL チューニング、SQL アクセス、メモリー、Undo およびセグメントなどのパフォーマンス・アドバイザは、システムのボトルネックを見つけ、候補となるソリューション・パス上の特定の領域に対するチューニング・アドバイスを得るために不可欠です。

語句の定義

データベース健全性監視インフラストラクチャは、この環境によく馴染みます。さらに詳細に説明する前に、まず、用語について理解しておきましょう。

サーバー生成アラート

サーバー生成アラートは、イベントまたはオブジェクトの状態に関する情報を含むメッセージで、急を要する問題を修正アクションの提案と共に通知します。通知を受け取ったユーザーは、提案されたアクションの記述に従って、レポートされた問題を解決できます。従来のアラートとの相違点は、SGA に直接アクセスすることにより、データベース自体が効率的でタイムリーにアラートを生成することです。

しきい値

アラートの多くは、計算済みの測定値に関連します。測定の境界値となるのがしきい値です。この境界を越えると、システムが望ましくない状態にあるとみなされます。しきい値は、さらに*アラート*（それほど深刻でない）および*危険*（業務に深刻な影響を及ぼすと思われるもの）に区分されます。指定された期間内*（監視期間）*に指定された回数*（連続出現）*だけ属性/測定の実績値がしきい値を越えると、アラートが発行されます。これにより、不適切なアラートを生成することなく、不定期に発生するスパイクを高度に処理できます。AWR スナップショットを使用して、適切なしきい値をシステムに設定できます。

アーキテクチャ

次に、サーバー生成アラートがどのように動作し、まだなぜそれを信頼できるかについて考えてみましょう。アラートというと、悪いことを連想します。いつも悪い知らせをもたらす人を、どうしたら好きになれるのでしょうか。タイムリーにトリガーされたアラートは、DBA の友人です。アラートは、問題が起こる前に予告するのです。「予知ではないが、限りなくそれに近いもの」なのです（大ヒット作品『*マイノリティ・レポート*』より）。

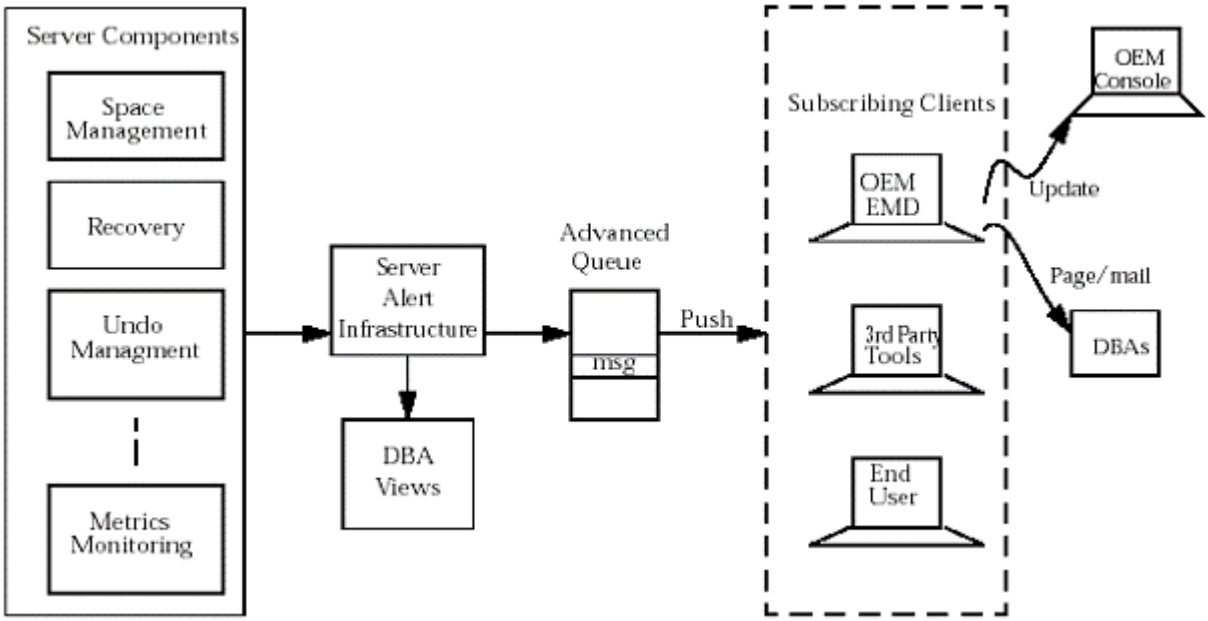

図 1

MMON の動作

MMON は毎分起動し、計測値を計算してイベントの発生をチェックします。測定に事前定義のしきい値、あるいはユーザーが定義したしきい値がある場合、MMON はそれらを実績値と比較して、超過している場合にはアラートを生成します。また特定のイベントが発生した場合にもアラートを生成します。

Oracle データベース・アラートには、アラートが生成されたオブジェクトの識別、問題の重大性、問題の説明、修正アクションに関するアドバイス、および場合によっては、さらに詳細なアドバイスを提供するアドバイザの名前が含まれています。表領域などのデータベース･エンティティはすべてオブジェクトです。

未処理のアラートは、SYS が所有する事前定義の永続詳細キュー（ALERT_QUE）に追加されます。アラートが発生すると、MMON はワークロード･リポジトリへの測定履歴のフラッシュを開始します。フラッシュは、アラートが発生した 1 時間前からアラートが解消された 2 分後まで続きます。この詳細情報は、推奨事項を生成し、適切なアドバイザを使用して問題を解決する上で欠かせません。

何らかの理由でアラートがアラート・キューに書き込まれない場合には、アラートに関する情報を含むメッセージがデータベース･アラート・ログに書き込まれます。記憶領域不足やメディア破損など特定のイベントが発生したときは、アラー

ト・キューに書き込まれません。その場合、データベース・アラート・ログが二次的なデータ永続メカニズムとして使用されます。

Enterprise Manager などアラート・キューのコンシューマ（*図1* 参照）が、未処理のアラートを配信します。設定によっては、管理者は、電子メールや文書を使用して管理コンソールで視覚的に通知を受けることができます。デフォルトでは、サーバー・アラートは常に Enterprise Manager Console で表示されます。サーバー生成アラートは、データベース健全性情報に関する真実の唯一のソースです。すべての情報が 1 つのソースから発信され、同じ情報を使用する複数のサブスクライバを追加の影響またはオーバーヘッドを与えずにサポートします。

未処理のアラートの重大性レベルは、定期的に評価されます。問題の現在のステータスが変化すると、更新後の重大度レベルが含まれる新しいアラート・メッセージが送信されます。問題状況が解消されると、未処理のアラートはアラート履歴に移されます。アラート履歴は、AWR スナップショット・パージ方針に従ってパージされます。デフォルトでは、すべてのアラート（解決済みと未解決の両方）が 7 日後にパージされます。

アラートの種類

サーバー生成アラートには、*しきい値ベースと非しきい値ベース*の 2 種類があります。

データベース生成アラートの多くはしきい値ベースであり、データベース測定のしきい値を設定する構成になっています。しきい値には、*アラート*しきい値と*危険*しきい値があります。アラートは、アラートしきい値をこえると*アラート・タイプ*に、危険しきい値を越えると*危険タイプ*に分類されます。

多くのしきい値はインスタンス固有であり、ユーザーによるカスタマイズが可能です。たとえば、*I/O 待機中のアクティブなセッション*、*CPU を使用中のアクティブなセッション*、*I/O イベント以外を待機中のアクティブなセッション*などです。中には、*表領域スペース使用*測定に基づくアラートなど、データベース全体のアラートもあります。これにより、Real Application Clusters（RAC）などの複数インスタンス環境で、重複したアラートの発行を防ぎます。

非しきい値ベースのアラートは、*スナップショットが古すぎる*、*再開可能なセッションが停止している*など、特定のデータベース・イベントに対応しています。この場合、アラートは単にイベントが発生したことを示します。

本番システム、設定およびワークロードは多様であるため、多くの場合、しきい値をデフォルトで設定しても意味をなしません。デフォルトでは、*表領域使用率*アラート（使用率 85%でアラート、97%で危険）、*スナップショットが古すぎる*、*リカバリ領域の空き領域不足*、および*再開可能なセッションが停止している*、というアラートが有効化されています。その他のアラートは、Enterprise Manager のアラートフレームワークによって複雑な設定なしに有効化されます。前述のように、Oracle では今後のリリースで、実際のシステム・ワークロードに基づいて意味のある適切なしきい値を数多く設定する予定です。

アラートメタデータの表示

前述のように、アラートはすべて ALERT_QUE というキューにパブリッシュされます。しきい値ベースのアラートは、DBA_OUTSTANDING_ALERTS ビューに表示され、解消されると DBA_ALERT_HISTORY 表に移されます。非しきい値ベースのアラートは、発行後、自動的に解消され、直接履歴表へ移動します。

システムレベル計測のインメモリー値は、V$SYSMETRIC ビューおよび V$SYSMETRIC_HISTORY ビューで表示できます。AWR への自動スナップショット収集の有効化だけで、オンディスク計測収集がすべての計測に対して有効化されます。計測のオンディスク値は、DBA_HIST_*ビューで表示できます。

サーバー・アラート情報の表示を可能にするその他のディクショナリ・ビューは、次のとおりです。

- DBA_THRESHOLDS: そのインスタンスに定義されるしきい値設定。
- V$ALERT_TYPES: 各アラート・タイプに関する情報。

Oracle Enterprise Manager 10*g* Database Control - **よいお知らせ**

停止時間やパフォーマンス低下を回避し、連続稼働するアプリケーションを完全にサポートするため、Enterprise Manager のアラート・インフラストラクチャおよびグラフィカル・ユーザー・インタフェースにより、ping、電子メールまたは SNMP による事前ユーザー通知が可能です。また管理者は、自動実行で問題を修正する応答アクションを各アラート状況に対して定義できます。EM では、特定のアラート状況が発生すると開始される応答アクションを作成できます。

次の使用例シナリオは、サーバー生成アラートと EM が急を要するパフォーマンス問題の洞察を提供する様子を示します。

a. データベース・ホーム・ページに、そのデータベースに対して生成されたアラートのおおまかな概要と、パフォーマンス・アドバイザによる診断結果が表示されます。待機率が 99%を越える待機ボトルネックに関する**危険**アラートがあります。また、ユーザー・コンテキスト依存のパフォーマンス・チューニング・アドバイスを提供する ADDM 診断結果もあります。図 2 にこれを示します。

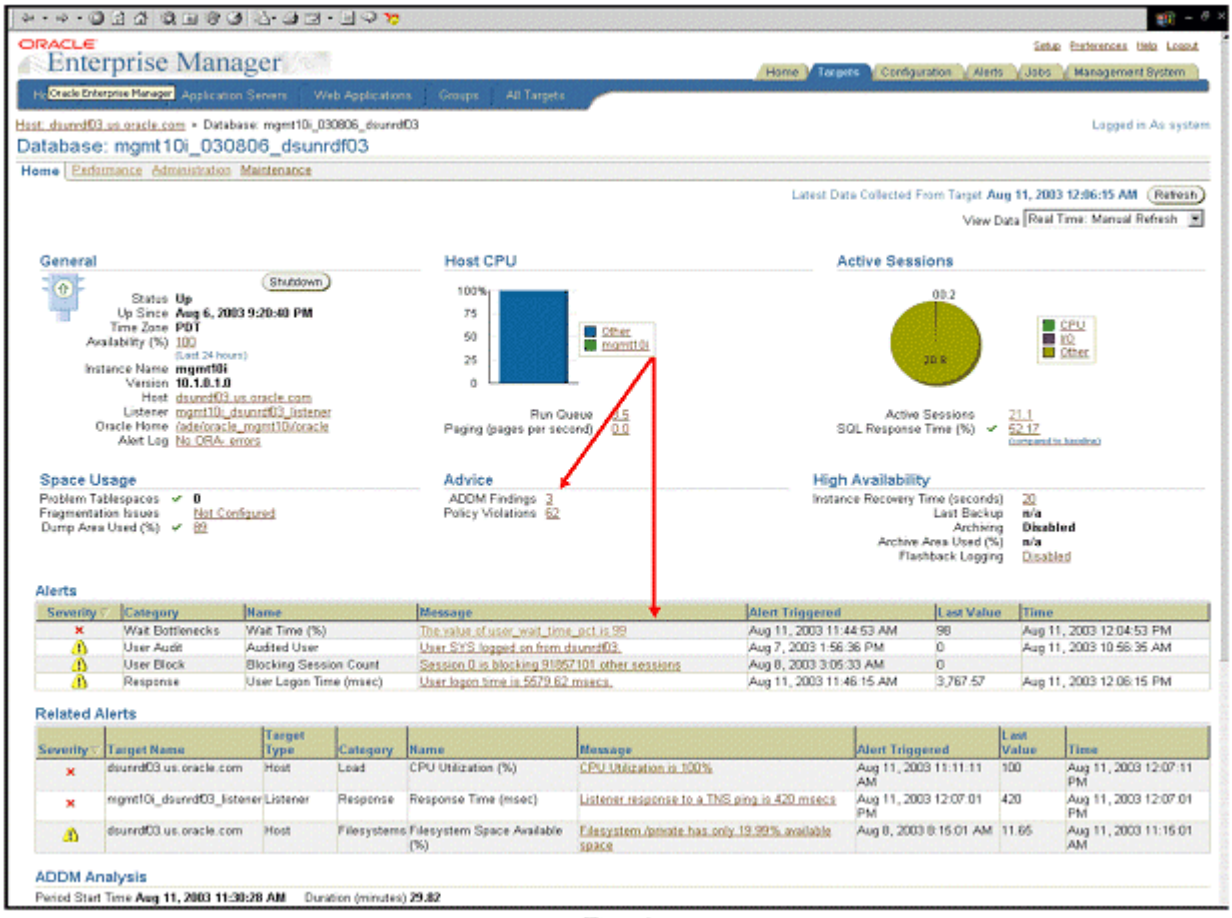

図 2

b. 「ADDM 診断結果」をクリックすると、「ADDM 診断結果詳細」ページが表示されます。このページから、問題の原因となった不正な SQL 文は DELETE であることがわかります。また、円グラフを見ると、ユーザーI/O 待機クラスがデータベース時間の 74%を使用していることが明確です。これらの診断結果は最新の ADDM 分析実行によるものです。ADDM 分析は、デフォルトで 30 分ごとに実行するようスケジュールされています。図 3 に追加情報が表示されます。

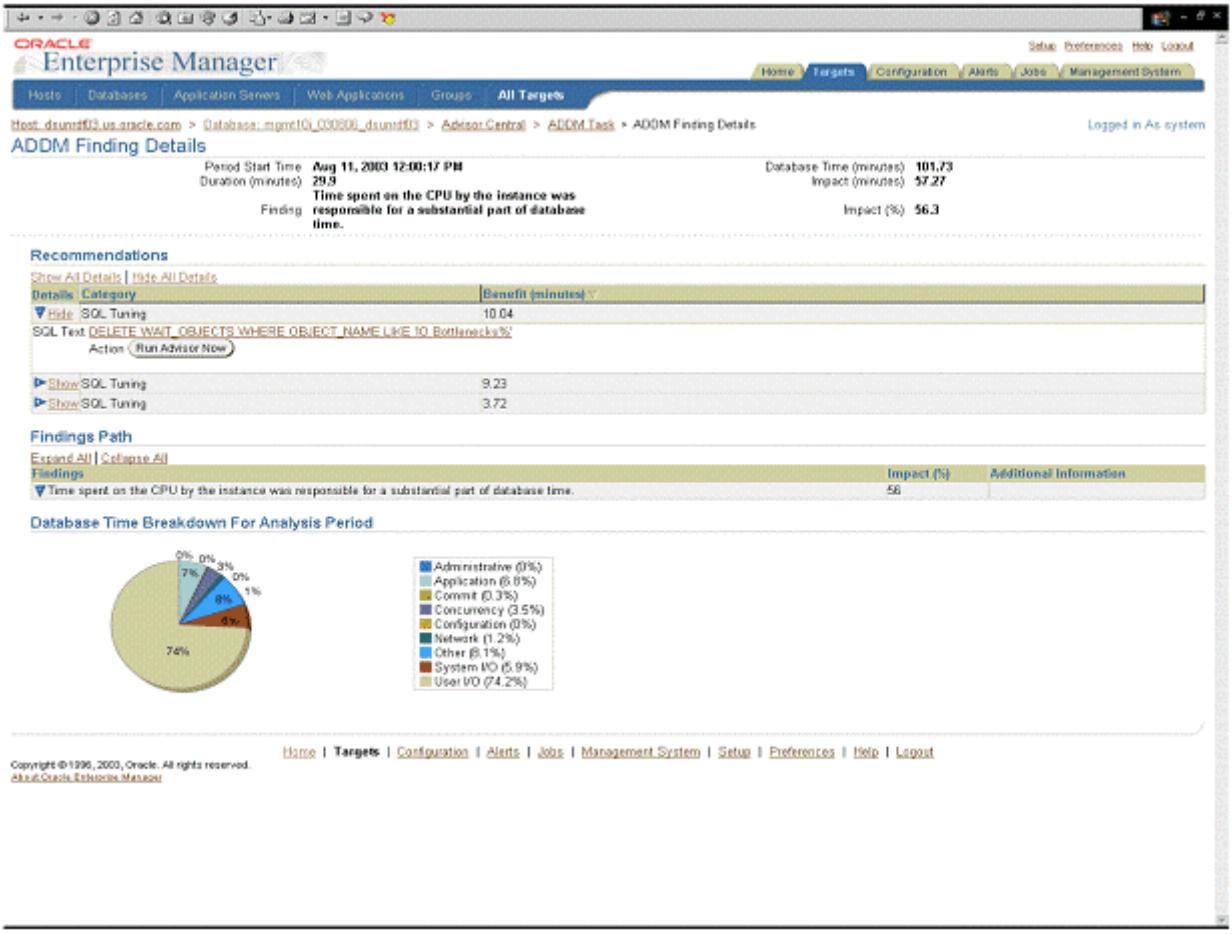

図 3

c. 現在起こっているパフォーマンス問題の判別には、「パフォーマンス」ページ（図 4）を表示します。これにより、データベースのアクティブなセッションに対する待機を構成する様々な待機クラスが明らかになります。この例では、待機クラス**ユーザーI/O** が待機の大半を占めていることがわかります。ホストの CPU リソースも掲載されています。

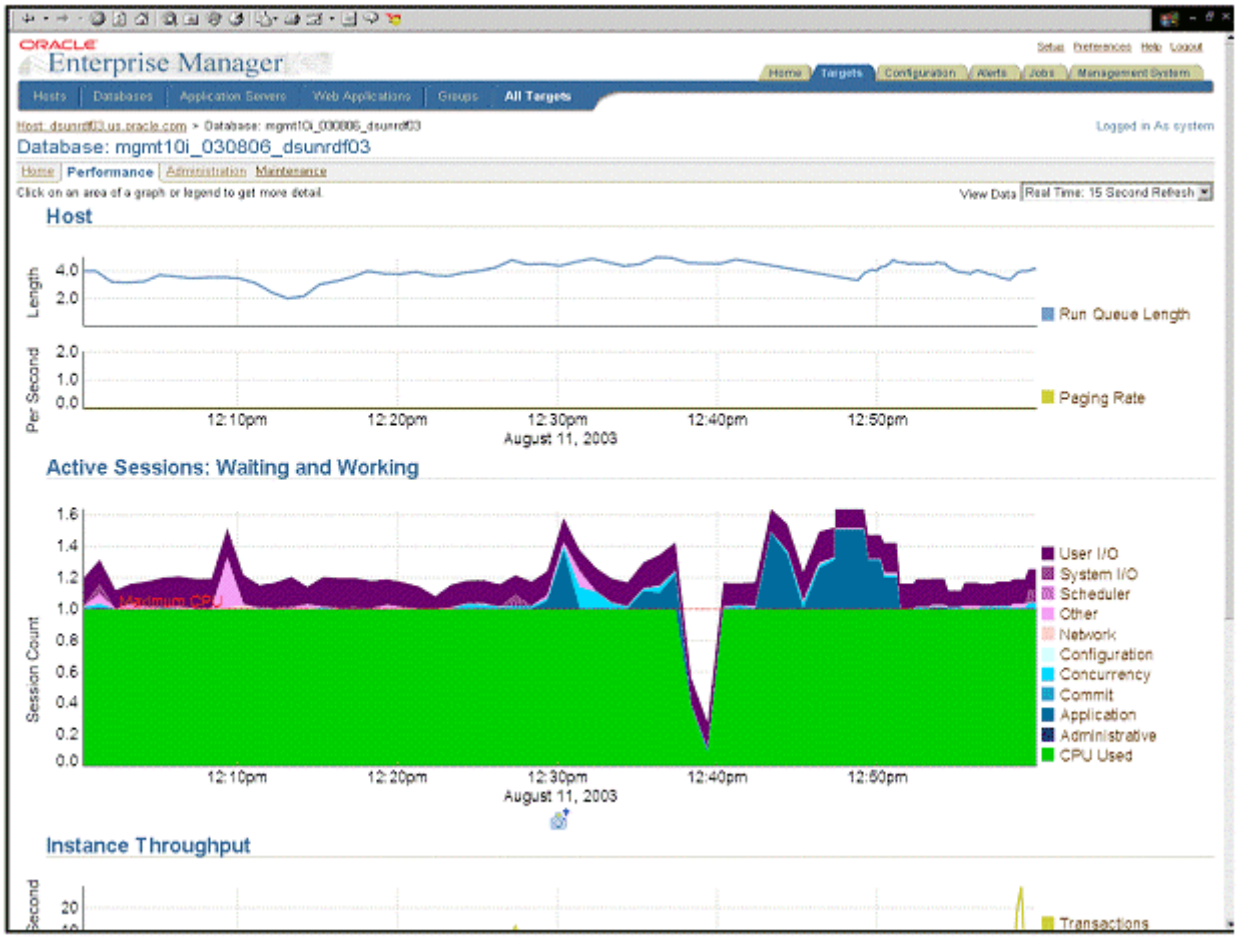

図 4

d. ユーザーI/O 凡例タグか紫のカラー帯（グラフの一番上）をクリックすると、ユーザーI/O の詳細が表示されます。ここでは、現在システムで問題となっている上位セッションおよび上位 SQL 文がわかります。さらに、図 5 に進みます。

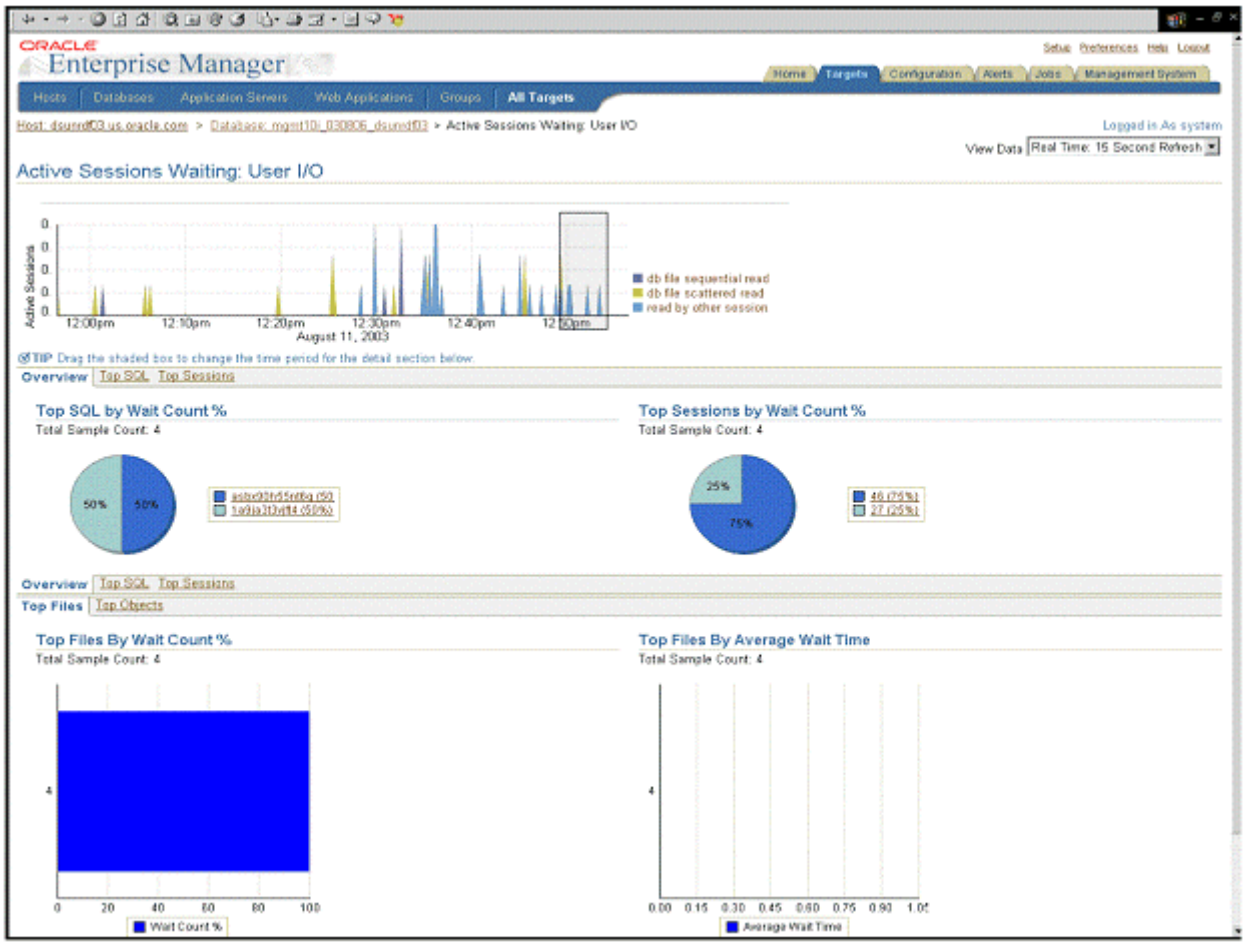

図 5

e. この問題の解決手順として、SQL チューニング･アドバイザの実行が推奨されます。図 3 の「推奨事項」を参照してください。ここには、SQL チューニング･アドバイザを DELETE に対して実行するボタンがあります。参考までに、問題解決方法として SQL チューニング･アドバイザでは、WAIT_OBJECTS.OBJECT_NAME 列に索引を作成することを推奨しました。

f. これで任務は完了です。サーバー生成アラートと EM Database Console の連携によりパフォーマンス問題の検出、正確な診断およびタイムリーな解決が実行できました。

しきい値の設定

Enterprise Manager のグラフィカル･インタフェースでは、必要な箇所にしきい値を簡単に設定し、管理できます。図 6 にその様子を示します。

○	User Limit Usage (%)	>	90	
○	User Logon Time (msec)	>	1000	
○	User Rollback Undo Records Applied (per second)	>		
○	User Rollback Undo Records Applied (per transaction)	>		
○	User Rollbacks (per second)	>		
○	User Rollbacks (per transaction)	>		
○	Wait Time (%)	>	50	70

図 6

サーバー生成アラート – PL/SQL API

もしあなたが根っからの開発者、つまりコマンドライン･インタフェース（CLO）の愛好家であり、EM がどのように、たやすく作業が済ませるかを知りたい場合には、興味を持ってこのセクションをお読みいただけると思います。ここには、サーバー生成アラートの使用にあたっての関連情報が、再利用可能なサンプル･コードを含むレシピと共に記載されています。

しきい値の管理

しきい値は一般に、稼働中のシステムを長期間観察し、各測定に対する適切な値の決定した後に設定します。Oracle Database10g のアラート・インフラストラクチャは、しきい値を変更する PL/SQL インタフェースを備えています。DBMS_SERVER_ALERT PL/SQL パッケージを使用して、現在設定されているしきい値の取得、および利用可能な測定への新しいしきい値の設定が可能です。このパッケージには次のプロシージャが含まれています。

- GET_THRESHOLD: 計測の設定を読み込みます。
- SET_THRESHOLD: 計測に新しいしきい値を定義します。

たとえば、表領域 KITCHEN のアラートしきい値を 60%、危険しきい値を 80%に設定するには、次のように記述します。

```
SQL> exec DBMS_SERVER_ALERT.SET_THRESHOLD(9000,DBMS_SERVER_ALERT.OPERATOR_GE,'60',
DBMS_SERVER_ALERT.OPERATOR_GE,'80',1,1,NULL,DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
'KITCHEN');
```

アラートアラートは、表領域 KITCHEN の使用率が 60%に達すると発行されます。同様に、危険アラートは、表領域 KITCHEN の使用率が 80%に達すると発行されます。

その他の入力パラメータは、次のように設定します。

- 外部測定識別子（この場合は 9000）。V$METRICNAME の METRIC_ID から取得できます。
- アラートしきい値の実績値を比較する演算子（*以上*あるいは*未満*など）。「以上」を定義する定数は、DBMS_SERVER_ALERT.OPERATOR_GE です。同様に、「未満」の定数は、DBMS_SERVER_ALERT.OPERATOR_LT です。
- この例では、アラートしきい値を 60 に設定します。
- 危険しきい値の実績値を比較する演算子。
- この例では、危険しきい値を 80 に設定します。
- 監視期間（分）。システムの実際の動作が何分間しきい値を越えたらアラートを発行するかを定義します。この規定により、短いスパイクに対してアラートが発行されることを防ぎます。この例では、監視期間は 1 です。
- 連続出現。測定値がしきい値を何度越えたらアラートを発行するかを定義します。この例では、連続出現は 1 に設定します。
- しきい値を設定するインスタンスの名前。シングル･インスタンス･データベースの場合には、値 NULL を使用します。
- オブジェクト･タイプ。しきい値を設定するオブジェクトのタイプです。この例では、定数 DBMS_ALERT.OBJECT_TYPE_TABLESPACE によって定義される表領域です。
- オブジェクトの名前。この例では、表領域名は KITCHEN です。

アラートの使用

発行されアラート・キューにパブリッシュされた Oracle アラートは、適切な権限を持つサブスクライバであれば誰でも利用できます。アラートを取得し、通知メカニズムで通知する一般的な手順は次のとおりです。下記に概説するレシピでは、ユーザーが SYS としてログインしており、SYSTEM データベース･ユーザーがアラートを使用するものとします。

1. ***ALERT_QUE のサブスクライブ***: DBMS_AQADM の実行権限を持つユーザーであれば誰でも、DBMS_AQADM.ADD_SUBSCRIBER プロシージャを使用して ALERT_QUE をサブスクライブできます。

```
/* The second argument of the following procedure is the subscribe, i.e., the agent on whose
behalf the subscription has been defined.
AQ$_AGENT プロシージャを使用してキューをサブスクライブします。サブスクライブするエージェントには、
ALERT_USR1 という名前が割り当てられています。 */
SQL> exec DBMS_AQADM.ADD_SUBSCRIBER(queue_name=>'ALERT_QUE',
AQ$_AGENT('ALERT_USR1','',0));
PL/SQL procedure successfully completed.

/* The "agent" in this context is the subscribing user */
SQL> exec DBMS_AQADM.CREATE_AQ_AGENT(agent_name=>'ALERT_USR1');
PL/SQL procedure successfully completed.
```

2. ***データベース・ユーザーとサブスクライブするエージェントとの関連付け***: ALERT_QUE はセキュアなキューであるので、サブスクライブするエージェントと関連付けられているユーザーのみがキュー内のメッセージにアクセスできます。DBMS_AQADM.ENABLE_DB_ACCESS プロシージャを使用してこの関連付けを行い、DBMS_ADQDM.GRANT_QUEUE_PRIVILEGE プロシージャを使用して実際の権限を割当てできます。

```
SQL> exec
DBMS_AQADM.ENABLE_DB_ACCESS(agent_name=>'ALERT_USR1',db_username=>'SYSTEM');
PL/SQL procedure successfully completed.

SQL> exec
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(privilege=>'DEQUEUE',queue_name=>'ALERT_QUE',\
grantee=>'SYSTEM',grant_option=>FALSE);
PL/SQL procedure successfully completed.
```

3. ***通知の登録***: 必要に応じて、アラートが ALERT_QUE キューに追加されたときにすぐに通知を受け取るよう登録できます。通知の形態は、電子メール、HTTP ポストまたは PL/SQL プロシージャです。通知の登録は、DBMS_AQ.REGISTER プロシージャを使用して行います。ALERT_QUE は永続キューです。そのため通知には、エンキュー・メッセージの AQ メタデータのみが含まれます。

```
SQL> DECLARE
       reginfo     aq$_reg_info;
       reginfolist aq$_reg_info_list;
     BEGIN
       reginfo := AQ$_REG_INFO('ALERT_QUE:ALERT_USR1',
       DBMS_AQ.NAMESPACE_AQ, 'mailto://janedoe@yourcompany.com',NULL);
       -- Create the registration info list
       reginfolist := AQ$_REG_INFO_LIST(reginfo);
       -- Register the registration info list
       DBMS_AQ.REGISTER(reginfolist, 1);
     END;
     /
PL/SQL procedure successfully completed.
```

4. ***電子メールと HTTP プロキシの構成***: エンキュー通知を電子メールまたは HTTP ポストを介して受け取る場合には、DBMS_AQELM を使用して、メール・ホスト名、メール・ポート番号、プロキシ・サーバー名など、メール・サーバーまたはプロキシ・サーバーに関する情報をデータベース・サーバーに設定が必要です。

```
SQL> BEGIN
       DBMS_AQELM.SET_MAILHOST('yourmailhost.com');
       DBMS_AQELM.SET_MAILPORT(25);
       DBMS_AQELM.SET_SENDFROM('janedoe@yourcompany.com');
       COMMIT;
     END;
PL/SQL procedure successfully completed.
```

5. *アラートのデキュー*: サブスクライバは、メッセージの内容を読むためにメッセージをデキューする必要があります。アラートをデキューするには、ALERT_QUE のデキュー権限が必要です。メッセージをデキューするには、DBMS_AQ.DEQUEUE プロシージャまたは OCIAQDeq コールを使用します。アラートメッセージは、サブスクライバがメッセージをデキューするか、有効期限までサブスクライバごとに ALERT_QUE に残ります。

```
SQL> DECLARE
       dequeue_options     dbms_aq.dequeue_options_t;
       message_properties  dbms_aq.message_properties_t;
       message             ALERT_TYPE;
       message_handle      RAW(16);
BEGIN
dequeue_options.consumer_name := 'ALERT_USR1';
  /* Never wait */
  dequeue_options.wait := DBMS_AQ.NO_WAIT;
  /* Always reset the position to the begining of the AQ */
  dequeue_options.navigation := DBMS_AQ.FIRST_MESSAGE;
  /* Remove the message when done */
  dequeue_options.dequeue_mode := DBMS_AQ.REMOVE;
  /* set some filter condition
  dequeue_options.deq_condition := 'tab.user_data.sequence_id > 6'; */
  DBMS_AQ.DEQUEUE(
            queue_name          =>    'ALERT_QUE',
            dequeue_options     =>    dequeue_options,
            message_properties  =>    message_properties,
            payload             =>    message,
            msgid               =>    message_handle);

  /* The alert message dequeued output presented below is a subset of the available
  fields in the message queue */
  DBMS_OUTPUT.PUT_LINE('Alert message dequeued:');
  DBMS_OUTPUT.PUT_LINE('  Timestamp:          ' || message.timestamp_originating);
  DBMS_OUTPUT.PUT_LINE('  Organization Id:    ' || message.organization_id);
  DBMS_OUTPUT.PUT_LINE('  Component Id:       ' || message.component_id);
  DBMS_OUTPUT.PUT_LINE('  Message Type:       ' || message.message_type);
  DBMS_OUTPUT.PUT_LINE('  Message Group:      ' || message.message_group);
  DBMS_OUTPUT.PUT_LINE('  Message Level:      ' || message.message_level);
  DBMS_OUTPUT.PUT_LINE('  Host Id:            ' || message.host_id);
  DBMS_OUTPUT.PUT_LINE('  Host Network Addr:  ' || message.host_nw_addr);
  DBMS_OUTPUT.PUT_LINE('  Reason:             ' ||

  DBMS_SERVER_ALERT.EXPAND_MESSAGE(userenv('LANGUAGE'),message.message_id,
  message.reason_argument_1,message.reason_argument_2,message.reason_argument_3,
  message.reason_argument_4,message.reason_argument_5));

  DBMS_OUTPUT.PUT_LINE('  Sequence Id:        ' || message.sequence_id);
  DBMS_OUTPUT.PUT_LINE('  Reason Id:          ' || message.reason_id);
  DBMS_OUTPUT.PUT_LINE('  Object Name:        ' || message.object_name);
  DBMS_OUTPUT.PUT_LINE('  Object Type:        ' || message.object_type);
  DBMS_OUTPUT.PUT_LINE('  Instance Name:      ' || message.instance_name);
  DBMS_OUTPUT.PUT_LINE('  Suggested action:   ' ||
  DBMS_SERVER_ALERT.EXPAND_MESSAGE(userenv('LANGUAGE'),
  message.suggested_action_msg_id,
  message.action_argument_1,message.action_argument_2,
  message.action_argument_3,message.action_argument_4, message.action_argument_5));
  DBMS_OUTPUT.PUT_LINE('  Advisor Name:       ' || message.advisor_name);
  DBMS_OUTPUT.PUT_LINE('  Scope:              ' || message.scope);
END;
/
PL/SQL procedure successfully completed.
```

```
/* The following output is formatted */

Alert message dequeued:
Timestamp:          08-AUG-03 13.43.15.392725 PM -07:00
Organization Id:    yourcompany.com
Message Type:       Notification
Message Group:      Configuration
Message Level:      32
Host Id:            acme
Host Network Addr: 555.555.555.555
Reason:             Threshold is updated on metrics "Tablespace Space Usage"
Sequence Id:        41
Reason Id:          137
Object Name:        KITCHEN
Object Type:        TABLESPACE
Instance Name:      inst1
Suggested action:   Check DBA_THRESHOLDS view to verify the result
Advisor Name:       Default Advisor
Scope:              Database
PL/SQL procedure successfully completed.
```

この PL/SQL の作業が煩わしいならば、いつでも EM に戻れます。EM では、これらすべてを舞台裏で実装し、ブラウザ・ベースのインタフェースで表示します。

結論

データベース健全性監視は、システム・パフォーマンス管理に欠かせないコンポーネントです。企業では、問題を早期に発見し、適切な診断および解決を行う必要があります。これまでは、過剰なアラートによって高いオーバーヘッドと不適切なアラートが発生し、多くのシステムに悪影響を与えていました。**サーバー生成アラート**では、観察者が観察者に与える影響を亜原子レベルに抑えていることから、**ハイゼンベルグの不確定原理**が忠実に描かれています。これは、観察者が観察対象の中に存在するためです。サードパーティの監視ソリューションでは、観察者が観察対象の外にいるため、不確定原理の影響が巨視的なレベルで現れます。

Oracle Database10g の組込み式サーバー生成アラートと Enterprise Manager の伝播フレームワークは、ブラウザ・ベースのインタフェースと共に、システム・パフォーマンス問題とデータベース・メンテナンス・タスクを管理する基盤となります。これにより、従来の監視システムと比較して、使用する金額とシステム・リソースの両面でコストが大幅に低減されます。結果として、Oracle Database10g が備える様々な自己管理機能により Oracle システムを自動化することで、手作業による介入の削減、コスト低減、および QoS の向上を実現します。

参照

1. Adams, D. *The Ultimate Hitchhiker's Guide to the Galaxy*. Del Rey, 2002. http://www.randomhouse.com

自己管理データベース: **自動健全性監視および変更**

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問合せ窓口:
電話: +1.650.506.7000
ファックス: +1.650.506.7200
www.oracle.com

オラクル社は、インターネット上での活動を強化するソフトウェアを提供します。

Oracle **はオラクル社の登録商標です。**
このガイドで使用されているさまざまな製品名およびサービス名には、オラクル社の商標が含まれています。
その他のすべての製品名およびサービス名は、各社の商標です。

この文書はあくまでも参考資料であり、掲載されている情報は予告なしに変更されることがあります。
万一、誤植などにお気づきの場合は、オラクル社までお知らせください。
オラクル社は本書の内容に関していかなる保証もしません。また、本書の内容に関連したいかなる損害についても責任を負いかねます。

Copyright © 2004 Oracle Corporation
All rights reserved.